AMENDMENTS TO THE CLAIMS

This listing of claims will replace all prior versions, and listings, of claims in the application:

Listing of Claims:

1   1. (Currently amended) A method for checkpointing an application,

2   comprising:

3   dynamically linking an interceptor library into the application <u>at</u>

4   <u>application startup time</u> during a run-time invocation of the application, wherein

5   the run-time invocation occurs after the application has been compiled and linked,

6   and wherein the interceptor library is dynamically linked by simply setting an

7   environment variable, without having to perform an entire static linking process;

8   intercepting a function call produced by the application at the interceptor

9   library;

10   recording parameters of the function call to create a checkpoint that

11   includes information about the function call parameters;

12   making the function call by referring to function pointers saved within the

13   interceptor library;

14   receiving results of the function call; and

15   forwarding results of the function call back to the application.

1   2. (Original) The method of claim 1, further comprising creating a

2   checkpoint by:

3   stopping the application;

4   retrieving the recorded parameters;

5   saving the checkpoint data, including the recorded parameters, to

6   secondary storage; and

7   resuming the application.

1   3. (Original) The method of claim 2, further comprising using the

2   checkpoint to restore the application.

1   4. (Original) The method of claim 2, wherein saving the checkpoint data to

2   secondary storage involves saving the checkpoint data to a persistent storage.

1   5. (Original) The method of claim 2, wherein saving the checkpoint data to

2   secondary storage involves saving the checkpoint data in a file system, or a

3   database.

1   6. (Original) The method of claim 1, wherein making the function call

2   involves referencing the function through a function pointer.

1   7. (Original) The method of claim 1, further comprising recording the

2   results of the function call to facilitate creating a checkpoint that includes

3   information about the results of the function call.

1   8. (Original) The method of claim 1, wherein the function calls can include

2   system calls or lib calls.

1   9. (Original) The method of claim 1, wherein the parameters can include:

2   file paths;

3   thread flags; and

4   timer-thread relationships.

1   10. (Currently amended) A computer-readable storage medium storing

2   instructions that when executed by a computer cause the computer to perform a

3   method for checkpointing an application, the method comprising:

4   dynamically linking an interceptor library into the application at

5   application startup time during a run-time invocation of the application, wherein

6   the run-time invocation occurs after the application has been compiled and linked,

7   and wherein the interceptor library is dynamically linked by simply setting an

8   environment variable, without having to perform an entire static linking process;

9   intercepting a function call produced by the application at the interceptor

10   library;

11   recording parameters of the function call to create a checkpoint that

12   includes information about the function call parameters;

13   making the function call by referring to function pointers saved within the

14   interceptor library;

15   receiving results of the function call; and

16   forwarding results of the function call back to the application.

1   11. (Original) The computer-readable storage medium of claim 10, further

2   comprising creating a checkpoint by:

3   stopping the application;

4   retrieving the recorded parameters;

5   saving the checkpoint data, including the recorded parameters, to

6   secondary storage; and

7   resuming the application.

1   12. (Original) The computer-readable storage medium of claim 11, further

2   comprising using the checkpoint to restore the application.

4

1   13. (Original) The computer-readable storage medium of claim 11,

2   wherein saving the checkpoint data to secondary storage involves saving the

3   checkpoint data to a persistent storage.

1   14. (Previously presented) The computer-readable storage medium of

2   claim 11, wherein saving the checkpoint data to secondary storage involves saving

3   the checkpoint data in a file system, or a database.

1   15. (Original) The computer-readable storage medium of claim 10,

2   wherein making the function call involves referencing the function through a

3   function pointer.

1   16. (Original) The computer-readable storage medium of claim 10,

2   wherein the method further comprises recording the results of the function call to

3   facilitate creating a checkpoint that includes information about the results of the

4   function call.

1   17. (Original) The computer-readable storage medium of claim 10,

2   wherein the function calls can include system calls or lib calls.

1   18. (Original) The computer-readable storage medium of claim 10,

2   wherein the parameters can include:

3   file paths;

4   thread flags; and

5   timer-thread relationships.

1   19. (Currently amended) An apparatus that checkpoints an application,

2   comprising:

5

3   a dynamic linking mechanism that is configured to dynamically link an

4   interceptor library into the application at application startup time during a run-

5   time invocation of the application, wherein the run-time invocation occurs after

6   the application has been compiled and linked, and wherein the interceptor library

7   is dynamically linked by simply setting an environment variable, without having

8   to perform an entire static linking process;

9   an intercepting mechanism within the interceptor library that is configured

10   to intercept a function call produced by the application;

11   a recording mechanism that is configured to record parameters of the

12   function call to facilitate creating a checkpoint that includes information about the

13   function call parameters;

14   a calling mechanism that is configured to make the function call by

15   referring to function pointers saved within the interceptor library;

16   a receiving mechanism that is configured to receive results of the function

17   call; and

18   a forwarding mechanism that is configured to forward results of the

19   function call back to the application.

1   20. (Original) The apparatus of claim 19, further comprising a checkpoint

2   creation mechanism that is configured to:

3   stop the application;

4   retrieve the recorded parameters;

5   save the checkpoint data, including the recorded parameters, to secondary

6   storage; and to

7   resume the application.

6

1   21. (Original) The apparatus of claim 20, further comprising a restoration
2   mechanism that is configured to use the checkpoint data to restore the application
3   to the checkpointed state.

1   22. (Original) The apparatus of claim 20, wherein the checkpoint creation
2   mechanism is configured to save checkpoint data to a persistent storage.

1   23. (Original) The apparatus of claim 20, wherein the checkpoint creation
2   mechanism is configured to save the checkpoint data in a file system, or a
3   database.

1   24. (Original) The apparatus of claim 19, wherein the calling mechanism
2   is configured to make the function call by referencing the function through a
3   function pointer.

1   25. (Original) The apparatus of claim 19, further comprising a recording
2   mechanism that is configured to record the results of the function call to facilitate
3   creating a checkpoint that includes information about the results of the function
4   call.

1   26. (Original) The apparatus of claim 19, wherein the function calls can
2   include system calls or lib calls.

1   27. (Original) The apparatus of claim 19, wherein the parameters can
2   include:
3   file paths;
4   thread flags; and
5   timer-thread relationships.

7